NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

TWO NEW APPROXIMATE SOLUTION TECHNIQUES FOR A MOVING TARGET PROBLEM WHEN SEARCHER MOTION IS CONSTRAINED

by

Metin Sagal

September 1985

Thesis Advisor: J. N. Eagle

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)*: Two New Approximate Solution Techniques For a Moving Target Problem When Searcher Motion is Constrained	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis, September 1985
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Metin Sagal	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943-5100	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943-5100	12. REPORT DATE: September 1985
	13. NUMBER OF PAGES: 43
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution is unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Moving target, detection, path, constraints

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The objective of this study is to develop and test two new approximate solution techniques for a moving target problem in discrete time and space where both the searcher and the target have constraints on their paths. The first technique is an application of the Local Search Method and the second technique is an application of the Frank-Wolfe Method. The motivation for looking at approximate methods is that the problem is NP-complete

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

and optimal solution techniques become impractical for large size problems. Experiments showed that the local search method approach is an efficient technique for obtaining approximate solutions. However, the Frank-Wolfe method approach does not perform well for the problem.

Two New Approximate Solution Techniques for a Moving Target Problem When Searcher Motion is Constrained.

by

Metin Sagal
Ltjg, Turkish Navy
B.S., Turkish Naval Academy, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

ABSTRACT

The objective of this study is to develop and test two new approximate solution techniques for a moving target problem in discrete time and space where both the searcher and the target have constraints on their paths. The first technique is an application of the Local Search Method and the second technique is an application of the Frank-Wolfe Method. The motivation for looking at approximate methods is that the problem is NP-complete and optimal solution techniques become impractical for large size problems. Experiments showed that the Local Search Method approach is an efficient technique for obtaining approximate solutions. However, the Frank-Wolfe Method approach does not perform well for the problem.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. THE PROBLEM

The problem considered here is the search for a moving target in discrete time and space, where there are constraints on the searcher's movement. The target is assumed to move among a finite set of cells $C=\{1,\ldots,N\}$ according to a specified Markov transition matrix $Z=\{z(i,j)\}$. The target's motion is independent of the searcher's actions. The transition matrix is known to the searcher as well as the initial distribution of the target over the search cells. In each time period one cell is searched. The searcher is able to move from his current cell, say i, to a cell which must be selected from a given subset C(i) of the set C. Thus, the search cell in a given time period must be within some specified neighborhood of the search cell in the previous time period. If the searcher and the target are in the same cell i, the target will be detected with probability q(i). If the target is not in the cell which is searched, it cannot be detected during the time period. The objective of the search is to find a T-time period search path which minimizes the probability of non-detection (PND).

B. BACKGROUND

The problem of searching for a moving target in discrete time and space has received considerable attention. The problem is difficult because of its complexity. Trummel and Weisinger [Ref. 1] proved that optimal constrained path search problem for a stationary target in discrete time and space where the time horizon is finite and the measure of

effectiveness is PND, is NP-Complete. Since the stationary target problem is NP-Complete, the moving target problem is also NP-Complete. For more information on NP-complete problems, see [Ref. 2]. The optimal solution techniques which have been used to solve the constrained path moving target problem are:

1. Total enumeration
2. The dynamic programming technique of Eagle [Ref. 3].

The total enumeration method is complete enumeration of all possible search paths. The disadvantage of this method is the exponential increase in computer time as the problem size increases. However, it has the advantage of requiring very little computer storage. Eagle formulates the problem as a Partially Observable Markov Decision Process (POMDP), and the method requires extensive computer storage, but finds solutions more quickly than the total enumeration method.

Since the problem is NP-Complete, it is difficult to solve large problems. For large problems, heuristic or approximate solution techniques have been proposed. Such methods were developed by Stewart [Ref. 4] and Eagle [Ref. 5].

Stewart developed a branch-and-bound algorithm for the solution of the problem. However, optimality cannot be guaranteed because exact bounds cannot be obtained. Nevertheless, Stewart used one-dimensional a random walk as a test problem and demonstrated that the algorithm performs efficiently.

Eagle's approximate procedure is a simpler version of his optimal dynamic programming method. Eagle used moving horizon policies and also looked for a lower bound for the minimal PND. Eagle's computational experience with

two-dimensional search problems showed the m-time period moving horizon policies (m-TPMH) do not necessarily perform better as m increases. Nonetheless, the method does show some promise. The m-TPMH policy selects as the next search that cell which would be optimal if m-time periods remained in the problem.

Recently Eagle and Yee developed a new approximate solution method by using the convex simplex method [Ref. 6]. Two-dimensional computational experiments showed that the method performs well.

In this thesis two new approximate solution techniques will be studied and tested. The first method is a heuristic approach which is an application of the Local (Neighborhood) Search. The second technique is an application of the Frank-Wolfe method.

II. FORMULATION OF THE PROBLEM

In this chapter we will formulate the problem as a nonlinear program.

A. DEFINITION OF SYMBOLS

T : Number of time periods.

N : Number of cells.

q(i) : Prob {detecting the target | target and searcher are in cell i}.

$Z = \{z(i,j)\}$ is an NxN Markov transition matrix where the (i,j)th element is the probability that the target in cell i at time period t will move to cell j at time period t+1.

$x(t) = \{x(1,t),.....,x(N,t)\}$ represents the searcher probability distribution at time t. That is, x(i,t) denctes the probability of the searcher being in cell i at time period t. At the start of the search, the searcher cell is assumed to be known with certainity, i.e., x(1) consists of 1 one and all others zero.

$P(t) = \{p(1,t),.....,p(N,t)\}$ represents the target probability distribution at time t. That is, p(i,t) is the probability target is in cell i at the begining of time period t without being detected by the previous t-1 searches. The initial target distribution P(1) is assumed to be known by the searcher.

R(i,j,t) is the probability that the target is in cell i without being detected by the previous t-1 searches and the searcher is in cell j.

$S(t) = \{S(i,j,t)\}$ is an NxN Markov transition matrix for the searcher. The (i,j)th element is zero if subset C(i) does not include j (path constraints).

$S = \{S(1),....,S(T-1)\}$ is called a search plan which includes all possible searcher motions for T-time period, given that the searcher starting cell is known.

B. NIP FORMULATION OF THE PROBLEM

Now we need to compute x(t)'s and R(i,j,t)'s. Since x(1) is known, we can determine x(t) by the Markovian motion as shown below:

$$x(t) = x(t-1)\ S(t-1), \quad t=2,...,T.$$

The R(i,j,t) consists of two Markov processes. Note that R(i,j,1) is assumed to be known. So, the R(i,j,t) can be computed as follows:

$$R(i,j,t) = \{1-q(i)\,d(i,j)\} \sum_{k,l} R(k,l,t-1)\,z(k,i)\,S(l,j,t-1)$$

$$i,j,k,l=1,2,\ldots,N \quad \text{and} \quad t=2,3,\ldots,T$$

where d(i,j) is one if i equals j, zero otherwise. Therefore, given P(1), q(i), S, and Z, the T-time probability of nondetection is

$$PND = \sum_{i,j} R(i,j,T), \quad i,j=1,2,\ldots,N.$$

We want to find the search plan S which minimizes the T-time period PND, subject to the path constraints. The final NLP form of the problem is given in Table 1. The following properties of the NLP were proven by Eagle and Yee [Ref. 6].

Proposition 1: The minimum of the NLP is achieved by a deterministic search plan. A deterministic search plan consists of ones and zeros.

Proposition 2: S is a deterministic search plan if and only if S is an extreme point of the linear constraints.

Proposition 3: The objective function is linear in S when constrained to any edge of the simplex.

TABLE 1

NLP FORMULATION OF THE PROBLEM

$$\min \quad PND$$

subject to:

$$S(t)\ 1 = 1 \quad , \quad t=1,\ldots,T-1$$

$$S(t) \geqslant 0 \quad , \quad t=1,\ldots,T-1$$

$$S(i,j,t) = 0 \quad , \quad t=1,\ldots,T-1 \quad i=1,\ldots,N \quad j \notin C(i)$$

III. THE LOCAL SEARCH ALGORITHM

The optimal solution techniques for hard combinatorial optimization problems require more computer time when the size of the problem increases. Therefore, they become difficult to implement for large size problems. One of the most succesful methods of attacking them is the Local Search or Neighborhood Search. However unsatisfying mathematically, this approach is certainly valid in practical situations. The main shortcoming of the method is that there is no criterion to tell whether the obtained solution is a global optimum. The local search method is most applicable for optimization in two main types of problems:

1. When the analytic relationship of the independent variables and the objective function is not known, but the value of the objective function can be evaluated at individual points by experiments.
2. When the analytic form of the objective function is known but there is no finite algorithm for obtaining the extreme value(s) in a closed form.

The efficiency of the method, relative to another, is defined in terms of a suitable cost function such as number of points or experiments required for localizing the optimum within a specified range. One of the best known illustrations of this technique is the traveling salesman problem (TSP) where the objective is to find the shortest path passing through each of a number of points exactly once and then returning to the starting point. For other examples refer to [Ref. 7].

Since the problem presented in the previous chapter is NP-complete and the current optimal solution techniques are impractical for large size problems, we can try to apply the

local search method as a heuristic approach for good approximate solutions.

A. THE ALGORITHM

First, we will define some concepts in order to introduce general local search algorithms and then apply this method to the constrained path moving target search problem.

Definition 1: An instance of an optimization problem is a pair (F,c) where F is any set, the domain of feasible points, and c is the objective function, a mapping

$$c: F \rightarrow R^1$$

Definition 2: Given an optimization problem with instances (F,c), a neighborhood is a mapping

$$N: F \rightarrow 2^F$$

defined for each instance.

The general local search algorithm is described as follows: Given an instance (F,c) of an optimization problem (minimizing) and an initial feasible point s, we search the neighborhood N(s) for an $m \in N(s)$ with $c(m) < c(s)$. So, we start with an initial feasible point s and attempt to improve the objective function by searching the neighborhood of s. If an improvement is found, another search is made in the neighborhood of the improved point. This continues until no improvement can be made.

Since we know that the solution of the NLP is an deterministic search plan (Proposition 1) and it is assumed that the searcher starting cell is known, we can reduce our feasible region to search paths $s=\{s(1),\ldots,s(T)\}$, where s(t) is the searcher cell at time period t. Therefore, we can start with an initial feasible search path s, an objective function value PND(s), and attempt to find a search path m in a neighborhood of s such that PND(m) < PND(s). We define the k-change neighborhood of search path s as all feasible search paths m which are identical to s except in k

consecutive time periods. So, the algorithm for the problem becomes:

Step 1. Start with an initial feasible search path s.

Step 2. Find the all possible k-change neighborhood paths by removing k consecutive cells from s (keeping the others unchanged) then replacing them with k new cells satisfying the path constraints.

Step 3. Find the k-change neighborhood path m with the minimum PND(m).

Step 4. If PND(m) < PND(s), set s=m and go to step 2. Otherwise, STOP; current s is k-optimal.

The most difficult task in any application is to decide of the proper value of k for the k-change policy. Selecting the right k leads to very effective heuristics for the TSP [Ref. 7]. We will test the present algorithm for k=1,2 and 3.

B. INITIAL FEASIBLE SEARCH PATH

The algorithm requires two stages. First, determine the initial feasible starting search path and then apply the procedure to it. The initial feasible point may be chosen randomly or it may be carefully constructed. In practice we need a quick and a good approximate solution. We do not want to be far away from the optimal when we start. In this study we will use the modified myopic policy (MMP) to determine an initial feasible starting path. The myopic policy, which gives optimal solution for a stationary target problems, always searches in the next time period t, the cell i which has the largest probability of detecting the target, $\{q(i)p(i,t)\}$. The MMP is defined as follows.

For the next time period t, searcher moves from his current cell i to that accessible cell j which has the largest $q(j)p(j,t)$ value. If all accessible cells j have identical values of $q(j)p(j,t)$, then the searcher will move to the accessible cell which has the minimum Euclidian

distance to the cell k having the largest $q(k)p(k,t)$ value. If the Euclidian distances are equal, these ties are broken randomly. There is probably a smarter technique for breaking ties.

The MMP is a heuristic construction meant to produce a good starting path, because the neighborhood k-change procedure may not be very powerful. We will also test the (k-1)-optimal solution as a starting path for the k-change policy.

IV. COMPUTATIONAL EXPERIENCE

Our computational experience is based on two-dimensional search problems. All experiments presented in this chapter were performed on an IBM 3033 mainframe computer using Fortran 77. The Local Search Algorithm was used to solve three 10-time period search problems, where N=9, 25 and 49. For each problem, values of k=1, 2 and 3 were used. The constraints on the searcher and the target paths are defined as; the searcher (target) is able to move to the cell previously searched (occupied) plus all adjacent cells. Cells are adjacent if they share a common side. For example, in Figure 4.1 the adjacent cells of cell 1, 4 and 5 are:

$$C(1)=\{1,2,4\} \ , \ C(4)=\{1,4,5,7\} \ , \ C(5)=\{2,4,5,6,8\}$$

The target transition matrix is; the target remains in the previously occupied cell with probability 0.4, and moves to the adjacent cells with probability 0.6/m where m is the number of adjacent cells. The probability of detection is 1.0 for each cell, i.e., $q(i)=1.0 \ , i=1,\ldots,N$.

A. 3 X 3 PROBLEM

1	2	3
4	5	6
7	8	9

Figure 4.1 9 Cell Search Grid.

In this problem the target moves among the 9 cells shown in Figure 4.1. The searcher starts in cell 1 and the target in cell 9, i.e., initial target distribution is that the target is in cell 9 with probability 1.0 and in the other cells with probabiliy 0. The 10-time period search results are shown in Table 2. An optimal solution was obtained for all k=1, 2, 3. However CPU time increases exponentially as k increases. As can be seen, the MMP solution is very close to the optimal. The Fortran code for this problem is given in Appendix A.

TABLE 2
3x3 PROBLEM RESULTS

	Search Path	PND	CPU Time(sec)
Myopic	4 5 8 5 6 5 8 5 6 5	0.2328	
k = 1	4 5 8 9 6 5 8 5 6 5	0.2214	0.51
k = 2	2 5 8 9 6 5 8 5 6 5	0.2214	3.10
k = 3	2 5 8 9 6 5 8 5 6 5	0.2214	26.18

OPTIMAL PND = 0.2214

B. 5 X 5 PROBLEM

In this example the target and the searcher moves among 25 cells (Figure 4.2). The searcher starts in cell 1 and the target starts in cell 13 with certainty. The 10-time period search results are shown in Table 3.

We found an optimal solution only for k=3, but solutions for k=1, 2 are close to the optimal. The k=2 solution

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure 4.2 25 Cell Search Grid.

TABLE 3
5x5 PROBLEM RESULTS

	Search Path	PND	CPU Time (sec)
Myopic	6 7 12 13 14 13 18 13 14 13	0.5099	
k = 1	6 7 12 13 14 19 18 13 8 9	0.4907	1.09
k = 2	2 7 12 13 8 13 18 19 14 9	0.4887	10.09
k = 3	6 7 8 13 12 17 18 19 14 9	0.4886	75.23

OPTIMAL PND = 0.4886

differs from the optimal solution only in the fourth digit. Similar to the 3x3 problem the MMP solution is close to the optimal and CPU time increases exponentially.

C. 7 X 7 PROBLEM

In this problem the target and the searcher moves among 49 search cells shown in Figure 4.3. The searcher starts in cell 1 and the target starts in cell 17. The 10-time period

search results are shown in Table 4. We obtained the optimal solution for only k=3. The percentage difference between the MMP solution and the optimal solution is only 1.2 %. For k=1 and 2 same result were obtained. Exponential increase in CPU time is more steep than the previous problems.

In addition, the total enumeration method was used to find all optimal paths for each problem. The results are shown in Table 4.

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42
43	44	45	46	47	48	49

Figure 4.3 49 Cell Search Grid.

In fact, for the 5x5 and 7x7 problems we increased the search area of the 3x3 problem with the same target and searcher starting cells and the target transition matrix. For each problem a similar pattern was observed for optimal paths. Specifically, the searcher goes to the target's starting cell as soon as possible, then starts to search outside of this cell similar to the expanding square search. Graphical representation of exponential increase in CPU time as k increases, is shown in Figure 5.1.

TABLE 4

7x7 PROBLEM RESULTS

	Search Path	PND	CPU Time(sec)
Myopic	2 9 10 17 24 17 16 17 10 11	0.5066	
k = 1	2 9 10 17 24 23 16 17 18 11	0.5063	1.83
k = 2	2 9 10 17 24 23 16 17 18 11	0.5063	9.55
k = 3	2 9 16 17 24 17 18 11 10 9	0.5005	117.73

OPTIMAL PND = 0.5005

TOTAL ENUMERATION RESULTS 1.

Problem	Possible Optimal Paths	Optimal PND	CPU Time(sec)
3x3	2 3 6 9 8 5 6 5 8 5 2 5 6 9 8 5 6 5 8 5 2 5 8 9 6 5 8 5 6 5 4 5 6 9 8 5 6 5 8 5 4 5 8 9 6 5 8 5 6 5 4 7 8 9 6 5 8 5 6 5	0.2214	240
5x5	6 11 12 13 8 9 14 19 18 17 6 11 12 13 18 19 14 9 8 7 6 7 8 13 12 17 18 19 14 9 6 7 8 13 14 19 18 17 12 7 6 7 12 13 8 9 14 19 18 17 6 7 12 13 18 19 14 9 8 7 2 7 12 13 8 9 14 19 18 17 2 7 12 13 18 19 14 9 8 7 2 7 8 13 12 17 18 19 14 9 2 7 8 13 14 19 18 17 12 7 2 3 8 13 12 17 18 19 14 9 2 3 8 13 14 19 18 17 12 7	0.4886	1656
7x7	8 9 10 17 18 17 24 23 16 9 8 9 16 17 24 17 18 11 10 9 8 15 16 17 24 17 18 11 10 9 2 3 10 17 18 17 24 23 16 9 2 9 10 17 18 17 24 23 16 9 2 9 16 17 24 17 18 11 10 9	0.5005	3260

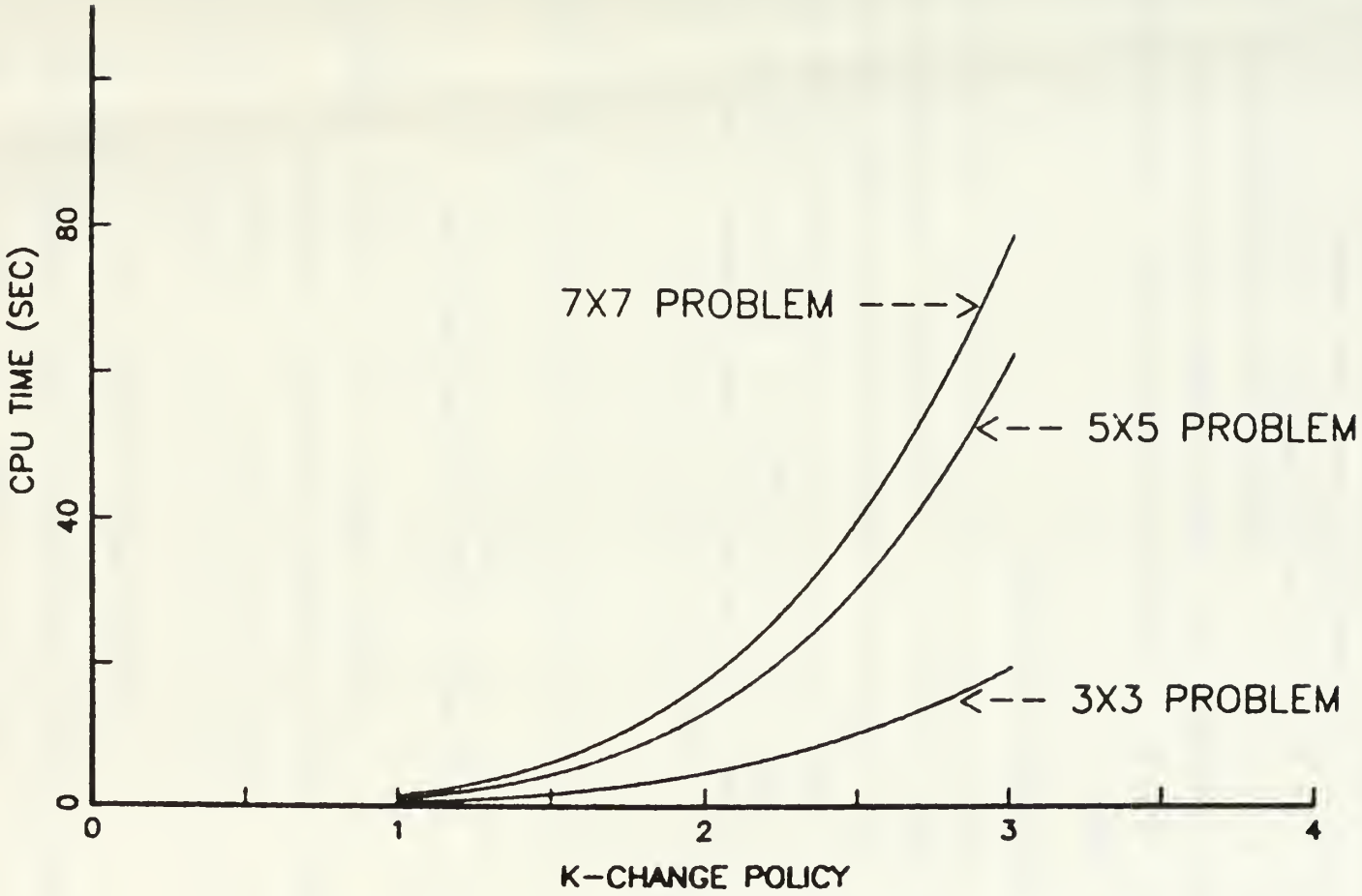

Figure 4.4 CPU Time vs k-change Policy.

D. MODIFIED MYOPIC SOLUTION RESULTS

The reason for using the MMP was to find a near optimal starting solution. This objective was met in the problems shown so far. In these problems, the mean position of the target does not move, given no search. In order to examine the possible effects of a moving mean target position, two 5x5 search problems were solved with different transition matrices, initial target distributions, and detection probabilities.

1. Fast Target Problem

In this problem the target starts in cell 13 and the searcher starts in cell 1. The target transition is such that the target stays in the previously occupied cell with probability 0 and moves to the adjacent cells with probability 1/m where m is the number of adjacent cells. The detection probability is 1.0 for each cell. The 10-time period search results are shown in Table 5.

TABLE 5
5x5 FAST TARGET PROBLEM RESULTS

	Search Path										PND	CPU Time (sec)
Myopic	2	7	8	8	13	18	13	18	13	8	0.4770	
k = 1	2	2	7	12	17	18	19	14	9	8	0.3742	1.8
k = 2	2	7	7	12	17	18	19	14	9	9	0.3742	10.19
k = 3	2	7	7	8	13	12	17	18	19	14	0.3779	67.93

OPTIMAL PND = 0.3742

In this case the MMP did not perform as well as it did in the previous 5x5 problems. An optimal solution was obtained for k=1, 2. Additionally, k=3 policy did not reach the optimal solution while requiring more CPU time.

2. Escaping Target Problem

For this problem, the target starts in cell 21. It then stays in the current cell with probability 0.1 and moves one cell up or to the right with probability 0.45. If it reaches one of the top boundary cells (1, 2, 3, 4) or one of the right boundary cells (25, 20, 15, 10), then it will move one cell to the right or one cell up with probability 0.9 and it remains in the current cell with probability 0.1. When it reaches cell 5 it will stay there forever and is assumed to have escaped. That is, the probability of detection in this cell is 0, while for the other cells the probability of detection is 1.0. The 10-time period search results are shown in Table 6. The MMP path has significantly greater PND than the optimal path even though it differs in only one time period from an optimal path. Optimal solutions were obtained for k=1, 2, 3.

For the last two problems all optimal paths were obtained by total enumeration and are shown in Table 6. In the fast target problem, the optimal search path placed search effort around the targets starting cell. If target moves in a specific direction (escaping target problem) this reduces the number of possible optimal paths. In this case, optimal paths put a barrier in front of the target.

E. INCREASING THE NUMBER OF TIME PERIODS

To examine the effect of increasing the number of time periods on CPU time, 3x3 problem was solved for 15, 20, 25 and 30 time periods. The graphical results are shown in

TABLE 6

ESCAPING TARGET PROBLEM RESULTS

	Search Path	PND	CPU Time (sec)
Myopic	6 11 12 12 13 14 15 15 10 10	0.2588	
k = 1	6 11 11 12 13 14 15 15 10 10	0.1530	0.68
k = 2	6 11 11 12 13 14 15 15 10 10	0.1530	3.03
k = 3	1 6 11 12 13 14 15 15 10 10	0.1530	18.46

OPTIMAL PND = 0.1530

TOTAL ENUMERATION RESULTS 2.

Problem	Possible Optimal Paths	Optimal PND	CPU Time (sec)
FAST TARGET	6 6 7 8 9 14 19 18 17 12 6 6 7 12 17 18 19 14 9 8 6 7 7 8 9 14 19 18 17 12 6 7 7 12 17 18 19 14 9 8 2 2 7 8 9 14 19 18 17 12 2 2 7 12 17 18 19 14 9 8 2 7 7 8 9 14 19 18 17 12 2 7 7 12 17 18 19 14 9 8 1 2 7 8 9 14 19 18 17 12 1 2 7 12 17 18 19 14 9 8 1 6 7 8 9 14 19 18 17 12 1 6 7 12 17 18 19 14 9 8	0.3742	1520
ESCAPING TARGET	6 6 11 12 13 14 15 15 10 10 6 11 11 12 13 14 15 15 10 10 1 6 11 12 13 14 15 15 10 10	0.1530	1524

Figure 4.5 for k=1, 2, and 3. For T=30 and k=3 program was stopped at 3600 seconds CPU time. The increse in CPU time for k=1 was considerably less than for k=2, 3. We can say that k=1 policy was the most CPU TIME / (1-PND) effective.

For the presented problems we also tested the (k-1)-change optimal solution as a starting path for the k-change policy where k=2. But starting with this alternative path did not improve the solution obtained by using a MMP starting path.

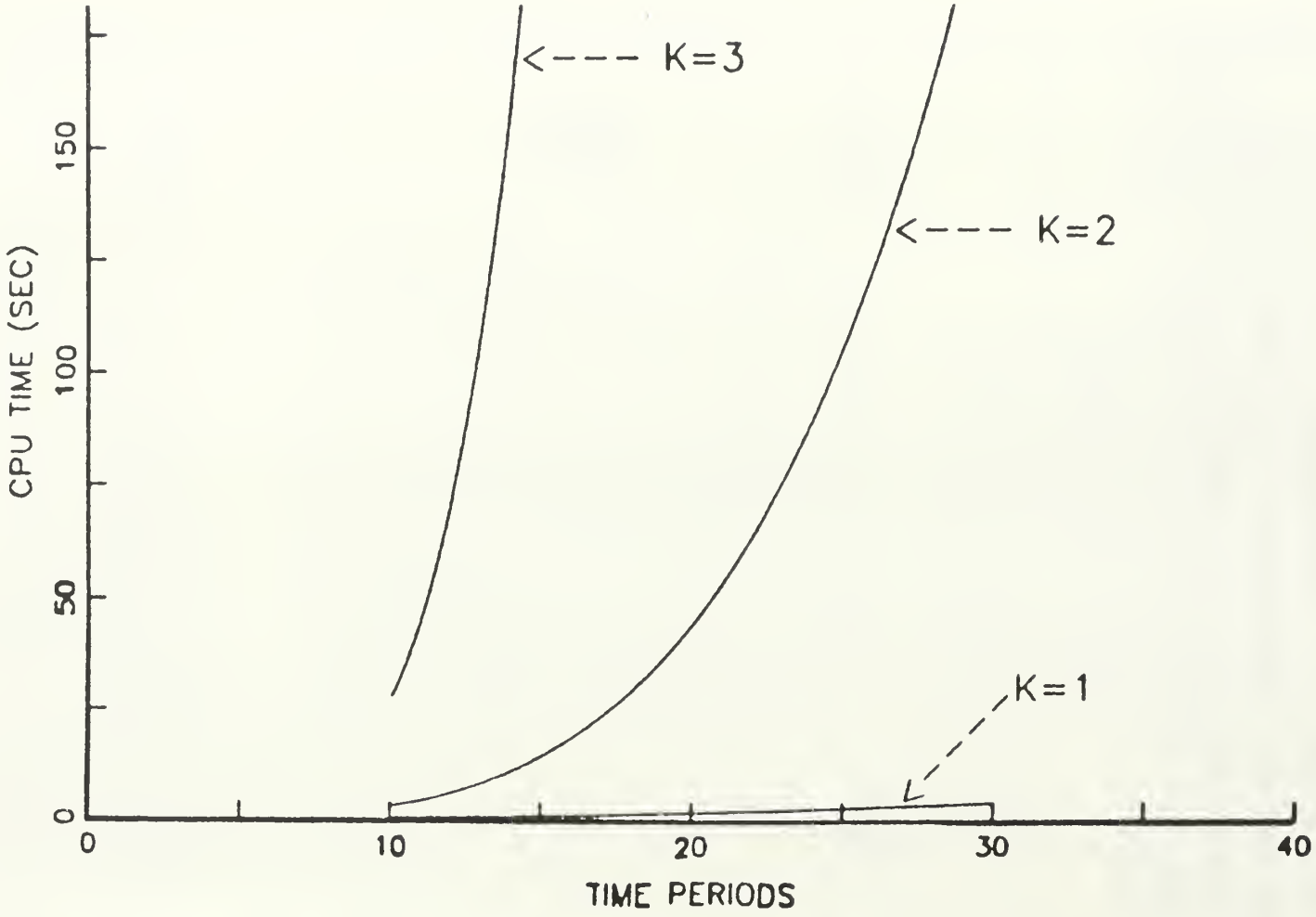

Figure 4.5 Time Periods vs CPU Time.

V. FRANK-WOLFE ALGORITHM

Another method applicable to the problem is the Frank-Wolfe (F-W) algorithm. The main motivation of selecting this algorithm is that this method implicitly examines all extreme points of the convex polyhedron, rather than just the k-change ones. In 1956 Marguerite Frank and Philip Wolfe proposed this method for solving non-linear programs having a convex differentiable objective function and linear constraints. Since our objective function is not convex this method will determine an approximate solution to the problem.

The F-W algorithm is a feasible direction method and may simply stated as follows; Given a feasible point at any iteration, an improving feasible direction is determined by linear program (LP). The objective function of the LP is formed by using a linear approximation (first order Taylor's expansion) of the objective function of the NLP and the constraints are the original linear constraints of the NLP. Then the step size is determined by a line search. The algorithm continues until no direction is found to improve the objective function. For further information refer to [Ref. 8].

A. APPLICATION OF F-W METHOD

Given S, x(t), P(t), q(i), and Z, the objective function can be written for each time period t as follows:

$$PND(S) = \sum_{i,j} p(i,t)\{1-q(i)x(i,t)\}z(i,j)v(j,t+1)$$

$$i,j=1,2,.....,N \quad \text{and} \quad t=1,2,.....,T$$

where $v(i,t)$ is the probability that the target is in cell i at the beginning of time period t and will not be detected by the remaining searches. We can compute $v(i,t)$'s in a backward recursive manner as follows:

$$v(i,T+1)=1.0\ ,\quad i=1,\ldots,N$$

$$v(i,t)=\sum_{j\in C(i)}\{1-q(i)\,x(i,t)\}z(i,j)\,v(j,t+1)$$

$$i=1,2,\ldots,N \quad \text{and} \quad t=1,2,\ldots,T$$

Since $P(1)$ is known, the $p(i,t)$'s can be computed by the forward recursion

$$p(j,t+1)=\sum_{i} p(i,t)\,\{1-q(i)\,x(i,t)\}z(i,j)$$

$$i,j=1,2,\ldots,N \quad \text{and} \quad t=1,2,\ldots,T-1$$

For a given search plan S the $p(i,t)$ and $v(i,t)$ will be the same for each time period. So, we need to compute them once for each search plan S.

Since $x(j,t)=x(i,t-1)S(i,j,t-1)$, then the gradient vector for time period t can be computed from the following:

$$Y(i,j,t)=\frac{\partial PND(s)}{\partial S(i,j,t)}=-x(i,t)\,p(j,t+1)\sum_{k\in C(j)} q(j)\,z(j,k)\,v(k,t+2).$$

$$i,j=1,2,\ldots,N \quad \text{and} \quad t=1,2,\ldots,T$$

Then the feasible direction of descent is found by solving the LP subproblem shown in Table 7. As we explained before the constraints of the LP must be the same as the NLP's constraints. Note that the LP's constraints in Table 7 is another way of expressing the constraints in Table 2. The solution of this LP subproblems is trivially found as follows; For each i and t, choose $j \in C(i)$ with minimum $Y(i,j,t)$, call it j^* then set $M(i,j^*,t)=1$ and $M(i,j,t)=0$ for $j \neq j^*$.

TABLE 7

LP SUBPROBLEM

$$\min \sum_{i,t} \sum_{j} Y(i,j,t)\ M(i,j,t)$$

subject to:

$$\sum_{j \in C(i)} M(i,j,t) = 1 \quad , \quad \begin{matrix} i=1,\ldots,N \\ t=1,\ldots,T-1 \end{matrix}$$

$$M(i,j,t) \geq 0 \quad , \quad \begin{matrix} i=1,\ldots,N \\ t=1,\ldots,T-1 \end{matrix}$$

This solution is an extreme point of the linear constraints. Therefore, solution of the LP is deterministic search plan M which defines a unique search path m. Since we are only interested in deterministic search paths, we choose the step size as 1 if $PND(m) < PND(s)$ and zero otherwise, ie., the process is terminated when $PND(m) > PND(s)$. So, the second algorithm becomes:

Step 1. Select an arbitrary feasible search path $s=\{s(1),..,s(T-1)\}$.

Step 2. Determine a new search path m by solving the LP subproblem.

Step 3. If PND(m) < PND(s) set s=m, go to step 1. Otherwise, STOP.

B. COMPUTATIONAL EXPERIENCE FOR F-W METHOD

We applied the algorithm to the 3x3 problem (defined in Chapter 4) with 30 randomly selected starting search paths (myopic search path included). The algorithm did not reach the optimal solution. For 7 of the 30 paths (myopic path was one of them) the algorithm did not improve the starting solution. The closest solution to optimal (0.2214) was 0.2328 which is the myopic path result. The maximum and minimum CPU times were 0.04 and 0.11 seconds, respectively. The results are shown graphically in Figure 4.4. The Fortran code of the algorithm for the 3x3 problem is given in Appendix B.

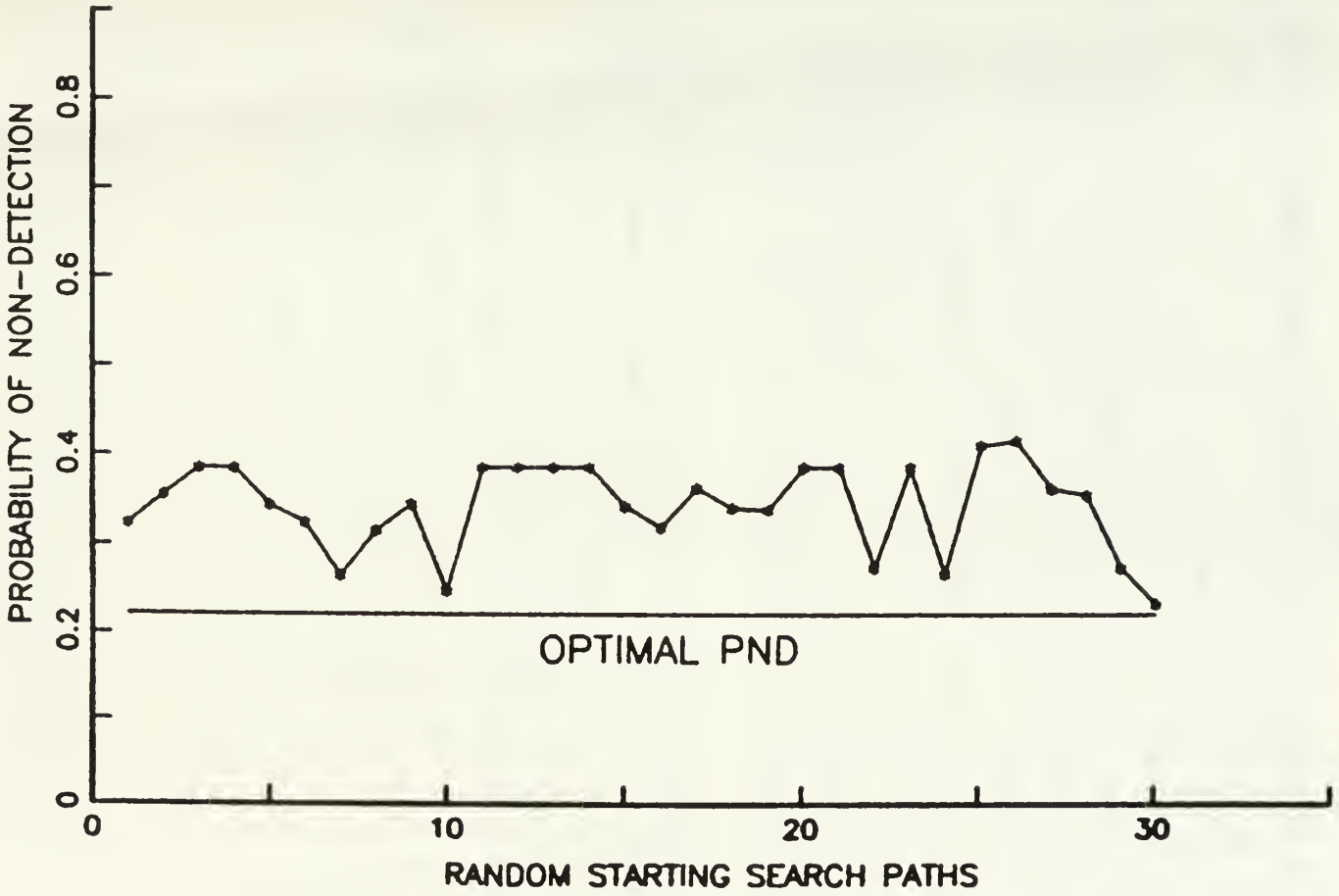

Figure 5.1 Results Of Frank-Wolfe Method.

VI. CONCLUSIONS

We have investigated the effectiveness and computational efficiency of the Local Search Method and the Frank-Wolfe Method for several two-dimensional search problems.

The heuristic Local Search Method appeared clearly satisfactory for the examined search problems. However, the exponential increase in CPU time will limit the use of the method for higher k values. Also increasing k does not necessarily improve the solution. This procedure appears to be useful because it allows a trade off between solution accuracy, i.e., closeness to the optimal search path and CPU time. For further analysis this method can be tested for different problems, k-change policies and number of time periods.

The Frank-Wolfe method approach did not perform well for the problems examined. An alternative way of improving efficiency of this algorithm may be to perform a line search after determining the feasible direction of descent. But this would increase the CPU time. Both methods are also applicable to Brown's original problem [Ref. 9] and they may be efficient for this problem because of its convex objective function.

APPENDIX A

FORTRAN CODE OF LOCAL SEARCH ALGORITHM

```
C    ***********************************************
C    *  METIN SAGAL                                *
C    *  THIS PROGRAM APPLIES LOCAL SEARCH          *
C    *  ALGORITHM TO 3X3 PROBLEM FOR K=1.          *
C    ***********************************************
C
      INTEGER T,NN,P(400,30),TT(50),S(50),C(25,25)
      INTEGER AA(9,5),IN(9)
      REAL PP(20),PB(50),BB(50),XMIN,PND
C
C    ***********************************************
C              * INPUT *
C      T=TIME PERIODS
C      NN=#OF SEARCH CELLS
C      S=STARTING SEARCH PATH
C      IN=#OF ADJACENT CELLS
C      A=ADJACENT CELLS
C      C=CONSTRAINT MATRIX
C    ***********************************************
C
      READ(5,100) T,NN
100   FORMAT(3I2)
      READ(5,103) (S(I),I=1,T)
103   FORMAT(11I1)
      READ(5,101) (IN(I),I=1,NN)
101   FORMAT(9I1)
      READ(5,102) AA(I,J),J=1,5),I=1,NN)
102   FORMAT(5I1)
      READ(5,104) ((C(I,J),J=1,NN),I=1,NN)
104   FORMAT(9I2)
C
C    ***** TARGET INITIAL PROB. DISTRIBUTION ******
C
      DO 15 K=1,NN
15      PB(K)=0
      PB(9)=1
      DO 25 K=1,NN
25    BB(K)=PB(K)
C
C    ********* FIND PND FOR STARTING PATH S *******
C
      DO 35 I=1,10
35      Z(I)=S(I+1)
      CALL PROB(Z,PB,T,NN,BB,PND)
      XMIN=PND
C
C    ********** FIND NEIGHBORHOOD PATHS ***********
C
999   MM=M=0
      TT=2
333   K=IN(S(TT-1))
      DO 10 I=1,K
         Z(TT-1)=AA(S(TT-1),I)
C
C    ********** CHECK PATH CONSTRAINTS ************
C
         IF(C(Z(TT-1),S(TT+1)) EQ.0) GO TO 10
C
C    ********** FIND PND FOR NEW PATH *************
C
         CALL PROB(Z,PB,T,NN,BB,PND)
```

```
         IF(PND.LT.XMIN) THEN DO
              M=M+1
              MM=MM+1
C
C    ********* KEEP THE BEST PATH AND PND *********
C
                XMIN=PND
                DO 30 IB=1,10
30            P(M,IB)=Z(IB)
                M=0
           END IF
           DO 40 IK=1,10
40         Z(IK)=S(IK+1)
10    CONTINUE
C
C    *********** CHANGE THE NEXT CELL **************
C
        TT=TT+1
        IF(TT.LE.11) GO TO 333
C
C    *** IF THERE IS AN IMPROVEMENT,THEN SET S ****
C   *TO THE BEST NEIGHBORHOOD PATH AND START AGAIN *
C
        IF(MM.GT.0) THEN DO
           S(1)=1
             DO 50 I=2,11
                S(I)=P(1,I-1)
50            Z(I-1)=P(1,I-1)
           GO TO 999
        END IF
C
C    ***** IF THERE IS NO IMPROVEMENT, STOP *******
C
        PRINT,'****** OPTIMAL PATH IS ******'
        WRITE(6,200) (S(I),I=2,11)
200     FORMAT(10I2)
        PRINT,'OPTIMAL PND IS',XMIN
        STOP
        END
C
C    ************** CALCULATES PND ****************
C
        SUBROUTINE PROB(Z,PB,T,NN,BB,PND)
        REAL A(50),PB(50),PND,BB(50)
        INTEGER NN,T,Z(50)
        KP=T-1
        DO 71 K=1,KP
        PB(S(K))=0
           A(1)=.4*PB(1)+.2*PB(2)+.2*PB(4)
           A(2)=.3*PB(1)+.4*PB(2)+.3*PB(3)+.15*PB(5)
           A(3)=.2*PB(2)+.4*PB(3)+.2*PB(6)
           A(4)=.3*PB(1)+.4*PB(4)+.15*PB(5)+.3*PB(7)
           A(5)=.2*PB(2)+.2*PB(4)+.4*PB(5)+.2*PB(6)+.2*PB(8)
           A(6)=.3*PB(3)+.15*PB(5)+.4*PB(6)+.3*PB(9)
           A(7)=.2*PB(4)+.4*PB(7)+.2*PB(8)
           A(8)=.15*PB(5)+.3*PB(7)+.4*PB(8)+.3*PB(9)
           A(9)=.2*PB(6)+.2*PB(8)+.4*PB(9)
                DO 72 1K=1,NN
                 PB(1K)=A(1K)
72               CONTINUE
71    CONTINUE
        PB(S(T))=0
        PND=0.0
        DO 51 K=1,NN
           PND=PND+PB(K)
51    CONTINUE
        DO 42 K=1,NN
           PB(K)=BB(K)
42    CONTINUE
```

```
      RETURN
      END
C
C    ********** INPUT DATA **********
C
15 9
125856585 65
343454343
124
1235
236
1457
24568
3569
478
5789
689
1  1  0  1  0  0  0  0  0
1  1  1  0  1  0  0  0  0
0  1  1  0  0  1  0  0  0
1  0  0  1  1  0  1  0  0
0  1  0  1  1  1  0  1  0
0  0  1  0  1  1  0  0  1
0  0  0  1  0  0  1  1  0
0  0  0  0  1  0  1  1  1
0  0  0  0  0  1  0  1  1
```

APPENDIX B

FORTRAN CODE OF F-W ALGORITHM

```
C
      INTEGER TIME,T,N,IN(9),A(9,9),S(10),CELL,Z(11)
      REAL R(9,9),X(9,10),V(9,11),PND,BB(9),PB(9)
      REAL C(9,10),XMIN,P(9,10)
C   ***********************************************
C        * INPUT *
C     T=TIME PERIODS
C     N=#OF CELLS
C     S=STARTING PATH
C     IN=#OF ADJACENT CELLS
C     A=ADJACENT CELLS
C     R=TRANSITION MATRIX
C   ***********************************************
C
      READ(5,100) T,N
 100   FORMAT(2I2)
      READ(5,101) (S(I),I=1,T)
 101   FORMAT(10I1)
      READ(5,102) (IN(I),I=1,9)
 102   FORMAT(9I1)
      READ(5,103) ((A(I,J),J=1,N),I=1,N)
 103   FORMAT(9I1)
      READ(5,104) ((R(I,J),J=1,N),I=1,N)
 104   FORMAT(9F3.2)
C
C   ********** DETERMINE V(I,J)'S **************
C
       CALL VFIND(S,T,N,V)
C
C   ********** DETERMINE P(I,J)'S **************
C
      CALL PFIND(S,T,P)
C
      DO 13 I=1,N
      PB(I)=P(I,1)
 13    BB(I)=P(I,1)
C
C   ********** DETERMINE X(I,J)'S **************
C
      CALL XFIND(S,T,N,X)
C
C   ******** FIND PND FOR A STARTING PATH ********
C
      CALL PROB(S,PB,T,N,BB,PND)
      XXMIN=PND
C
C   ********* FIND NEW EXTREME POINT *************
C
      Z(1)=1
 999  CALL CFIND(2,Z(1),IN,X,R,V,A,P,CELL)
         Z(2)=CELL
      CALL CFIND(3,Z(2),IN,X,R,V,A,P,CELL)
         Z(3)=CELL
      CALL CFIND(3,Z(3),IN,X,R,V,A,P,CELL)
         Z(4)=CELL
      CALL CFIND(4,Z(4),IN,X,R,V,A,P,CELL)
```

```
         Z(5)=CELL
       CALL CFIND(5,Z(5),IN,X,R,V,A,P,CELL)
         Z(6)=CELL
       CALL CFIND(6,Z(6),IN,X,R,V,A,P,CELL)
         Z(7)=CELL
       CALL CFIND(7,Z(7),IN,X,R,V,A,P,CELL)
         Z(8)=CELL
       CALL CFIND(8,Z(8),IN,X,R,V,A,P,CELL)
         Z(9)=CELL
       CALL CFIND(9,Z(9),IN,X,R,V,A,P,CELL)
         Z(10)=CELL
       CALL CFIND(10,Z(10),IN,X,R,V,A,P,CELL)
         Z(11)=CELL
C
C    ****** FIND PND FOR A NEW EXTREME POINT ******
C
       CALL PROB(Z,PB,T,N,BB,PND)
C
C    ******* COMPARE TWO EXTREME POINT'S PND ******
C
       IF(PND.LT.XXMIN) THEN DO
       XXMIN=PND
       DO 90 I=1,T
 90        S(I)=Z(I+1)
C
C    * DETERMINE NEW V(I,J)'S, P(I,J)'S, X(I,J)'S *
C
       CALL VFIND(S,T,N,V)
       CALL PFIND(S,T,P)
       DO 11 I=1,N
       PB(I)=P(I,1)
 11     BB(I)=P(I,1)
       CALL XFIND(S,T,N,X)
       GO TO 999
       END IF
C
C    ** IF THE NEW EXTREME POINT DOES NOT IMPROVE **
C    ********* THE OBJECTIVE FUNCTION, STOP *********
C
 500   PRINT,'MIN PND IS',XXMIN
       PRINT,'****** OPTIMAL PATH IS ******'
       WRITE(6,600) (S(I),I=1,T)
 600    FORMAT(3X,10I2)
       STOP
       END
C
C    ************ CALCULATES V(I,J)'S ************
C
       SUBROUTINE VFIND(S,T,N,V)
       REAL V(9,11)
       INTEGER S(10),T
       DO 15 I=1,N
 15        V(I,11)=1.0
       J=T
 25     K=J+1
       V(1,J) = .4*V(1,K)+.3*V(2,K)+.3*V(4,K)
       V(2,J) = .2*V(1,K)+.4*V(2,K)+.2*V(3,K)+.2*V(5,K)
       V(3,J) = .3*V(2,K)+.4*V(3,K)+.3*V(6,K)
       V(4,J) = .2*V(1,K)+.4*V(4,K)+.2*V(5,K)+.2*V(7,K)
       V(5,J) = .15*V(2,K)+.15*V(4,K)+.4*V(5,K)
       V(5,J) = V(5,J)*(.15*V(6,K)+.15*V(8,K))
       V(6,J) = .2*V(3,K)+.2*V(5,K)+.4*V(6,K)+.2*V(9,K)
       V(7,J) = .3*V(4,K)+.4*V(7,K)+.3*V(8,K)
       V(8,J) = .2*V(5,K)+.2*V(7,K)+.4*V(8,K)+.2*V(9,K)
       V(9,J) = .3*V(6,K)+.3*V(8,K)+.4*V(9,K)
       V(S(J),J)=0
       J=J-1
       IF(J.GT.0) GO TO 25
       RETURN
```

```
      END
C
C    ****** CALCULATES COST COEFFICIENTS AND ******
C    ********** SOLVES LP SUBPROBLEM **********
C
      SUBROUTINE CFIND(TIME,MM,IN,X,R,V,A,P,CELL)
       REAL R(9,9),V(9,11),P(9,10),PP,CMIN,X(9,10),C(9,10)
      INTEGER TTIME,TIME,MM,IN(9),A(9,9),CELL,MI,MK(10),LL
      CMIN=1
      IK=IN(MM)
      TTIME=TIME+1
      DO 40 I=1,IK
        J=A(MM,I)
        PP=0.0
        IB=IN(J)
         IF(P(J,TIME).EQ.0) THEN DO
            DO 55 II=1,IB
              PP=PP+R(J,A(J,II))*V(A(J,II),TTIME)
 55         CONTINUE
            C(MM,J)=-(0.0001)*PP
         ELSE DO
            DO 56 II=1,IB
              PP=PP+R(J,A(J,II))*V(A(J,II),TTIME)
 56         CONTINUE
            C(MM,J)=-(P(J,TIME)*PP)
         END IF
      IF(C(MM,J).LT.CMIN) THEN DO
         CMIN=C(MM,J)
         CELL=J
      END IF
 40   CONTINUE
      RETURN
      END
C
C    ********** CALCULATES X(I,J)'S **********
C
      SUBROUTINE XFIND(S,T,N,X)
      INTEGER S(10),T
      REAL X(9,10)
      DO 60 I=1,T
         DO 70 J=1,N
         X(J,I)=0.0
 70    CONTINUE
 60    CONTINUE
      DO 78 I=1,T
 78       X(S(I),I)=1.0
      RETURN
      END
C
C    ********** CALCULATES PND **********
C
      SUBROUTINE PROB(S,PB,T,N,BB,PND)
      REAL AA(50),PB(9),PND,BB(9)
      INTEGER N,T,S(10)
      KP=T-1
      DO 71 K=1,KP
         PB(S(K))=0
         AA(1)=.4*PB(1)+.2*PB(2)+.2*PB(4)
         AA(2)=.3*PB(1)+.4*PB(2)+.3*PB(3)+.15*PB(5)
         AA(3)=.2*PB(2)+.4*PB(3)+.2*PB(6)
         AA(4)=.3*PB(1)+.4*PB(4)+.15*PB(5)+.3*PB(7)
         AA(5)=.2*PB(2)+.2*PB(4)+.4*PB(5)+.2*PB(6)+.2*PB(8)
         AA(6)=.3*PB(3)+.15*PB(5)+.4*PB(6)+.3*PB(9)
         AA(7)=.2*PB(4)+.4*PB(7)+.2*PB(8)
         AA(8)=.15*PB(5)+.3*PB(7)+.4*PB(8)+.3*PB(9)
         AA(9)=.2*PB(6)+.2*PB(8)+.4*PB(9)
              DO 72 IK=1,N
              PB(IK)=AA(IK)
 72           CONTINUE
```

```
 71    CONTINUE
          PB(S(T))=0
       PND=0.0
       DO 50 K=1,N
          PND=PND+PB(K)
 50    CONTINUE
       DO 42 K=1,N
          PB(K)=BB(K)
 42    CONTINUE
       RETURN
       END
C
C    ************ CALCULATES P(I,J)'S **************
C
       SUBROUTINE PFIND(S,T,P)
       REAL P(9,10),PM(9,10)
       INTEGER S(10),T,M
       P(9,1)=1.0
       PM(9,1)=1.0
       DO 85 I=1,8
          PM(I,1)=0
 85       P(I,1)=0
       DO 80 J=2,T
       M=J-1
       PM(S(M),M)=0.0
         P(1,J)=.4*PM(1,M)+.2*PM(2,M)+.2*PM(4,M)
         P(2,J)=.3*PM(1,M)+.4*PM(2,M)+.3*PM(3,M)+.15*PM(5,M)
         P(3,J)=.2*PM(2,M)+.4*PM(3,M)+.2*PM(6,M)
         P(4,J)=.3*PM(1,M)+.4*PM(4,M)+.15*PM(5,M)+.3*PM(7,M)
         P(5,J)=.2*PM(2,M)+.2*PM(4,M)+.4*PM(5,M)+.2*PM(6,M)
         P(5,J)=P(5,J)*.2*PM(8,M)
         P(6,J)=.3*PM(3,M)+.15*PM(5,M)+.4*PM(6,M)+.3*PM(9,M)
         P(7,J)=.2*PM(4,M)+.4*PM(7,M)+.2*PM(8,M)
         P(8,J)=.15*PM(5,M)+.3*PM(7,M)+.4*PM(8,M)+.3*PM(9,M)
         P(9,J)=.2*PM(6,M)+.2*PM(8,M)+.4*PM(9,M)
       DO 86 I=1,9
 86    PM(I,J)=P(I,J)
 80    CONTINUE
       RETURN
       END
C
C    ****************** INPUT DATA *****************
C
10 9
4565856585
343454343
124000000
123500000
236000000
145700000
245680000
356900000
478000000
578900000
689000000
040030000030000000000000000
020040020000020000000000000
000030040000000030000000000
020000000040020000020000000
000015000015040015000015000
000000020000020040000000020
000000000030000000040030000
000000000000020000020040020
000000000000000030000030040
```

LIST OF REFERENCES

1. Trummel, K. E., and Weisinger, J. R., The Complexity of the Optimal Searcher Path Problem, Daniel H. Wagner Associates, 1983.

2. Aho, A. V., Hopcroft, J. E., Ullman, J. D., The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

3. Eagle, J. N., "The Optimal Search for a Moving Target When The Searcher Path is Constrained", Operations Research, v. 32, pp. 1107-1115, September 1983.

4. Stewart, T. J., "Search for a Moving Target When Searcher Motion is Restricted", Computers and Operations Research, v. 6, pp. 129-140, 1979.

5. Naval Postgraduate School, Report No: NPS 55-84-009, The Approximate Solution of a Simple Constrained Search Path Moving Target Problem Using Moving Horizon Policies, by Eagle, J. N., 1984.

6. Eagle J. N., Yee, J. R., An Approximate Solution Technique for the Constrained Search Path Moving Target Problem, Naval Postgraduate School, Submitted for Publication.

7. Papadimitriou, C. H., Steiglitz, K., Combinatorial Optimization, Algorithms and Complexity, Prentice-Hall, 1982.

8. Wagner, H. M., Principles Of Operations Research , Prentice-Hall, 1975.

9. Brown, S. S., "Optimal Search for a Moving Target in Discrete Time and Space", Operations Research, v. 28, pp. 1275-1289, December 1980.

INITIAL DISTRIBUTION LIST

No.		Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Professor J. N. Eagle, Code 55ER Naval Postgraduate School Monterey, California 93943-5100	1
4.	Professor J. R. Yee, Code 55YE Naval Postgraduate School Monterey, California 93943-5100	1
5.	Deniz Kuvvetleri Komutanligi Bakanliklar, Ankara/TURKEY	5
6.	Bogazici Universitesi Yoneylem Arastirmasi Bolumu Istanbul/TURKEY	1
7.	Ortadogu Teknik Universitesi Yoneylem Arastirmasi Bolumu Ankara/TURKEY	1
8.	Metin SAGAL Sumer sok. Akbank Sitesi 1.Blok Daire 13. Goztepe, Istanbul/TURKEY	1

Thesis
S152157 Sagal
c.1
Two new approximate solution techniques for a moving target problem when searcher motion is constrained.